

CSCE 471/871 Lecture 5: Building Phylogenetic Trees

Stephen D. Scott

1

Phylogenetic Trees

- Assumption: all organisms on Earth have a common ancestor

⇒ all species are related in some way

- Relationships represented by *phylogenetic trees*
- Trees can represent relationships between orthologs or paralogs
 - Orthologs: genes in different species that evolved from a common ancestral gene by speciation (evolution of one species out of another). Normally, orthologs retain the same function in the course of evolution.
 - Paralogs: genes related by duplication within a genome. In contrast to orthologs, paralogs evolve new functions

3

Phylogenetic Trees (cont'd)

- This is a weighted tree, where each weight ("edge length") is an estimate of evolutionary time between events
 - Based on distance measure (e.g. substitution scoring matrices) between sequences
 - Gives a reasonably accurate approximation of relative evolutionary times, despite the fact that sequences can evolve at different rates
- Number of possible binary trees on n nodes grows exponentially in n
 - E.g. $n = 20$ has about 2.2×10^{20} trees
 - We'll use heuristics, of course

5

Outline

- **Phylogenetic trees**
- Building trees from pairwise distances
- Parsimony
- Simultaneous sequence alignment and phylogeny

2

Phylogenetic Trees (cont'd)

- We'll use binary trees, both rooted and unrooted
 - Rooted for when we know the direction of evolution (i.e. the common ancestor)
 - Can sometimes find the root by adding a distantly related organism/sequence to an existing tree (Figure 7.1, page 162)

4

Outline

- Phylogenetic trees
- **Building trees from pairwise distances**
 - Distance measures
 - UPGMA
 - The ultrametric property of distances
 - Additivity and neighbor joining
- Parsimony
- Simultaneous sequence alignment and phylogeny

6

Building Trees from Pairwise Distances

UPGMA

- Start with some distance measure between sequences
 - E.g., **Jukes-Cantor**: $d_{ij} = -0.75 \log(1 - 4f_{ij}/3)$, where f_{ij} is fraction of residues that differ between sequences x_i and x_j when pairwise aligned
- UPGMA (unweighted pair group method average) algorithm
 - One of a family of **hierarchical clustering algorithms**
 - Basic overall idea of this algorithmic family: Find minimum inter-cluster distance d_{ij} in current distance matrix, merge clusters i and j , then update distance matrix
 - Differences among algorithms lie in matrix update
 - For phylogenetic trees, also add edge lengths

7

Building Trees from Pairwise Distances

UPGMA (cont'd)

- $\forall i$, assign seq x_i to cluster C_i and give it its own leaf, with height 0
- While there are more than two clusters
 - Find minimum d_{ij} in distance matrix
 - Add to the clustering cluster $C_k = C_i \cup C_j$ and delete C_i and C_j
 - For each cluster $C_\ell \notin \{C_k, C_i, C_j\}$

$$d_{k\ell} = \frac{1}{|C_k| + |C_\ell|} \sum_{p \in C_k, q \in C_\ell} d_{pq}$$

[Shortcut: Eq. (7.2)]

- Add to the tree node k with children i and j , with height $d_{ij}/2$

- When only C_i and C_j remain, place root at height $d_{ij}/2$

Example: Fig 7.4, page 168

8

Building Trees from Pairwise Distances

UPGMA (cont'd)

- If the rate of evolution is the same at all points in original (target) phylogenetic tree, then UPGMA will recover the correct tree
 - This occurs iff length of all paths from root to leaves are equal in terms of evolutionary time
- If this is not the case, then UPGMA may find incorrect topology (Fig. 7.5, p. 170)
- Can avoid this if distances satisfy **ultrametric** condition: for any three sequences x_i, x_j, x_k , the distances d_{ij}, d_{jk}, d_{ik} are either all equal, or two are equal and one is smaller

9

Building Trees from Pairwise Distances

Neighbor Joining

- If the ultrametric property doesn't hold, can still recover original tree if **additivity** holds
 - I.e. if, in the original tree, the distance between any pair of leaves = the sum of the lengths of the edges of the path connecting them
- If additivity holds, then **neighbor joining** finds the original tree
 - First, find a pair of neighboring leaves i and j , assign them parent k , then replace i and j with k , where for all other leaves m ,

$$d_{km} = (d_{im} + d_{jm} - d_{ij})/2$$

- But it does **NOT** work to simply choose the pair (i, j) with minimum d_{ij} (See Fig. 7.7, p. 171)
- Instead, choose (i, j) minimizing $D_{ij} = d_{ij} - (r_i + r_j)$, where L is current set of "leaves" and

$$r_i = \frac{1}{|L| - 2} \sum_{k \in L} d_{ik}$$

10

Building Trees from Pairwise Distances

Neighbor Joining (cont'd)

- Initialize $L = T =$ set of leaves
- While $|L| > 2$
 - Choose i and j minimizing D_{ij}
 - Define new node k and set $d_{km} = (d_{im} + d_{jm} - d_{ij})/2$ for all $m \in L$
 - Add k to T with edges of lengths $d_{ik} = (d_{ij} + r_i - r_j)/2$ and $d_{jk} = d_{ij} - d_{ik}$
 - Update $L = \{k\} \cup L \setminus \{i, j\}$
- Add final, length- d_{ij} edge between final nodes i and j

11

Outline

- Phylogenetic trees
- Building trees from pairwise distances
 - Parsimony**
 - Weighted parsimony
 - Score computation
 - Branch and bound
- Simultaneous sequence alignment and phylogeny

12

Parsimony

- Very widely used approach for tree building
 - Scores a tree based on the cost of substitutions in going from a node to its child
- ⇒ will assign hypothetical ancestral sequences to internal nodes
- Example, page 174 (unit costs)
 - Generally consists of two components
 1. Computing cost of tree T over n aligned sequences
 2. Searching through the space of possible trees for min-cost one
 - Treat each site independently of the others, so for a length- m alignment, run the scoring algorithm on each of the m sites separately
 - Let $S(a, b)$ be cost of substituting b for a
 - Scoring site (tree) $u \in \{1, \dots, m\}$, let $S_k(a)$ be the minimal cost for the assignment of symbol (residue) a to node k

13

Parsimony Scoring a Tree

1. Initialize $k = 2n - 1$ (index of the root node)
2. Recursively compute $S_k(a)$ for all a in the alphabet:
 - (a) If k is a leaf, set $S_k(a) = 0$ for $a = x_u^k$ and $S_k(a) = \infty$ otherwise
⇒ a must match u th symbol in sequence
 - (b) Else $S_k(a) = \min_b(S_i(b) + S(a, b)) + \min_b(S_j(b) + S(a, b))$, where i and j are k 's children
3. Return $\min_a S_{2n-1}(a)$ as minimum cost of tree

Can recover ancestral residues by tracking where min comes from in recursive step

14

Parsimony Searching for a Tree

- Not practical to enumerate the entire set of possible trees and score them all
- Will use **branch and bound** to speed it up (though no guarantee of an efficient algorithm)
 - When incrementally building a tree, adding edges will never decrease its cost
 - Thus if a tree's cost already exceeds the final cost of the best tree so far, we can discard it
- Algorithm: systematically grow existing tree by adding edges, stopping expansion if current tree's cost exceeds final cost of best tree so far

15

Outline

- Phylogenetic trees
- Building trees from pairwise distances
- Parsimony
- **Simultaneous sequence alignment and phylogeny**
 - **Hein's affine cost algorithm**

16

Simultaneous sequence alignment and phylogeny Hein's Affine Cost Algorithm

- Similar to parsimony in that, given a topology, it infers ancestral sequences
- But this algorithm uses an affine gap penalty model (separate penalties for opening and extending gaps)
- First, it ascends the tree from the leaves, determining the set of sequences that best align with leaf sequences
 - Represents such a set of sequences as a digraph
- Then it works its way up toward the root, at each step inferring the set of sequences that best align with the child graphs
- Finally, it descends from the root to the leaves, fixing the specific ancestral sequences

17

Simultaneous sequence alignment and phylogeny Hein's Affine Cost Algorithm

Finding Set of Sequences that Best Align with Leaves

- **GOAL:** Given sequences x and y , find set of sequences such that for each such sequence z , $S(x, z) + S(z, y) = S(x, y)$ (for either mismatch scores or weighted scores)
- Use dynamic programming to handle affine gap penalties, avoiding alternating gaps:
 - $V^M(i, j) = \text{min cost aligning } x_{1..i} \text{ to } y_{1..j} \text{ with } x_i \text{ aligned to } y_j$

$$V^M(i, j) = \min\{V^M(i-1, j-1), V^X(i-1, j-1), V^Y(i-1, j-1)\} + S(x_i, y_j)$$
 - $V^X(i, j) = \text{min cost aligning } x_{1..i} \text{ to } y_{1..j} \text{ with } x_i \text{ aligned to gap}$

$$V^X(i, j) = \min\{V^M(i-1, j) + d, V^X(i-1, j) + e\}$$
 - $V^Y(i, j) = \text{min cost aligning } x_{1..i} \text{ to } y_{1..j} \text{ with } y_j \text{ aligned to gap}$

$$V^Y(i, j) = \min\{V^M(i, j-1) + d, V^Y(i, j-1) + e\}$$

18

Simultaneous sequence alignment and phylogeny

Hein's Affine Cost Algorithm

Finding Set of Sequences that Best Align with Leaves (cont'd)

- Dynamic programming example in Fig. 7.13, page 183
 - j indexes rows, i indexes columns; seq. X is bottom/horizontal
 - Where do the X values in column 2 come from? Y values in row 2? (Hint: what do these values represent?)
- Result is a set of paths through the DP table, each corresponding to a valid ancestral sequence
 - If one spot on a path is a match between x_i and y_j , then a valid ancestral sequence contains either x_i or y_j in that position
 - If a gap is involved, then can take the gap or the residue
 - * But since cost function is not linear, need to either take the entire gap or none of the gap
 - * E.g. in Fig. 7.13, with leaves CAC and CTCACA, can use as ancestral sequence CTC, CAC, CACACA, etc., but not CACAC (why?)

19

Simultaneous sequence alignment and phylogeny

Hein's Affine Cost Algorithm

Finding Set of Sequences that Best Align with Leaves (cont'd)

- Can represent set of sequences as a digraph (e.g. Fig. 7.14(a); edges directed to the right), aka a sequence graph, where each path through the graph corresponds to a valid ancestral sequence
- Null ("dummy") edges (denoted by δ) allow gaps to be entirely skipped

20

Simultaneous sequence alignment and phylogeny

Hein's Affine Cost Algorithm

Building Sequence Graphs for Higher-Level Nodes

- Now want to ascend the tree towards the root, building ancestral sequence graphs for internal nodes
- But SG construction previously described ran DP on individual sequences!
- Turns out we can also run DP on SGs
 - In DP equations, " $i - 1$ " means the set of previous nodes in the horizontal graph, " $j - 1$ " the set of previous nodes in the vertical graph
 - Now take minimum over entire set of previous nodes that have values defined (non-" $-$ ")
 - Scoring function S now defined on sets; it's 0 iff its set-type arguments have non-empty intersection
- Once DP completed, do another traceback and build new SG
 - When labeling edges in new SG, use the intersection of the labels in the two defining edges, or the union if the intersection is empty

21

Simultaneous sequence alignment and phylogeny

Hein's Affine Cost Algorithm

Filling in Ancestral Sequences

- Now choose a path in the root's SG, then go to child nodes and trace their SGs with its parent's ancestral seq., choosing compatible symbols
- In final multiple alignment, need to fill in gaps

22

Simultaneous sequence alignment and phylogeny

Hein's Affine Cost Algorithm

Building the Topology

- Still need to build the tree to align sequences to
- Hein's tree-building algorithm:
 1. Compute an informative subset of the inter-sequence distances
 2. Build a "distance tree" by adding sequences to it one by one
 3. Perform rearrangements on the tree to improve its fit to the distance data
 4. Align sequences to the tree (what we already covered)

23

Simultaneous sequence alignment and phylogeny

Hein's Affine Cost Algorithm

Building the Topology

Computing Subset of Distances

- Assume that the distance measure and sequences form a metric space, implying:
 - $d(s_1, s_2) = 0 \Leftrightarrow s_1 = s_2$
 - $d(s_1, s_2) = d(s_2, s_1)$
 - $d(s_1, s) + d(s, s_2) \geq d(s_1, s_2)$
- Can use third eq. to upper- and lower-bound unknown distances
- I.e. if differences between upper and lower bounds is smaller than a parameter, do not compute the exact value

24

Simultaneous sequence alignment and phylogeny

Hein's Affine Cost Algorithm
Building the Topology
Computing Distance Tree

- Add sequences one at a time
- Choose to add to T_{k-1} the sequence s_k minimizing

$$d(s_k, T_{k-1}) = \min_{s_j \in \text{leaves}(T_{k-1})} \{d(s_k, s_j)\}$$

25

Simultaneous sequence alignment and phylogeny

Hein's Affine Cost Algorithm
Building the Topology
Computing Distance Tree (cont'd)

- Choose s_k 's attachment point as follows:
 - Let s_1 be sequence in tree most similar to s_k
 - Let A be internal node closest to s_1 , and $S = \{t_1, t_2, t_3\}$ be the set of subtrees leaving A
 - For each $t_i, t_j \in S$, compute $d(t_i, s_k)$ and $d(t_i, t_j)$
 - Hypothetically, if we attached s_k on the path from t_i to t_j , then we'd place it at point p_{ij} such that

$$d(t_i, p_{ij}) = (d(t_i, s_k) + d(t_i, t_j) - d(t_j, s_k))/2$$

26

Simultaneous sequence alignment and phylogeny

Hein's Affine Cost Algorithm
Building the Topology
Computing Distance Tree (cont'd)

- – Now let $v_1 = \text{avg distance in direction of } t_1 \text{ of } p_{12} \text{ and } p_{13} \text{ from } A$; similarly define v_2 and v_3
- Maximum of these 3 distances determines attachment point
(Intuition: If t_i far from A and near s_k , this is s_k 's home)
- If the max v_i takes us past the root of t_i , then t_i 's root becomes A and the process repeats
- Once all nodes added, look at interchanging neighbors in tree to improve score

Topic summary due in 1 week!

27